Deterministic Simulation

What modern online games can learn from the Game Boy

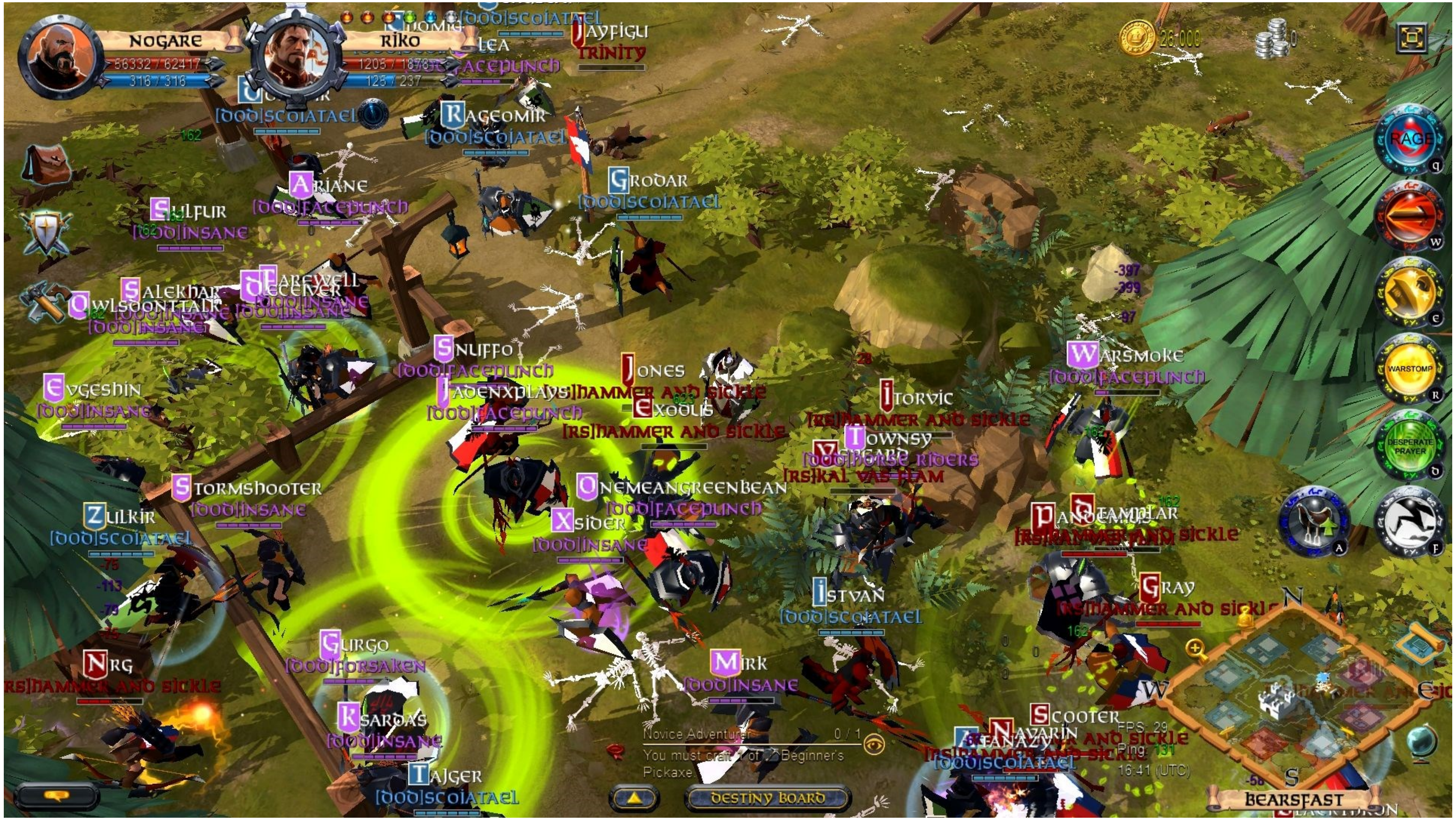
David Salz

CTO, Sandbox Interactive

Who am I?

- CTO, Co-Founder of Sandbox Interactive
- 15 years in the industry

- Albion Online:
  - Sandbox MMORPG
  - Cross-Platform (Windows/OSX/Linux/Android/iOS)
  - Player-Driven Economy (everything is player-crafted)
  - Strong focus on PvP + Guilds
  - Currently in Beta w/ 120.000+ „founding“ players
  - Using Unity Engine



Nogare
56332 / 62417
316 / 316
Riko
1205 / 1878
125 / 237
26,000
RAGE
WARSTOMP
DESPERATE PRAYER
Ariane
[DOD]FACEPUNCH
Sulfur
[DOD]INSANE
Ragecomir
[DOD]SCOIATAEL
Grodar
[DOD]SCOIATAEL
Jayfigu
TRINITY
Snuffo
[DOD]FACEPUNCH
Jones
[RS]HAMMER AND SICKLE
Exodus
Torvic
Warsmoke
[DOD]FACEPUNCH
Evgeshin
[DOD]INSANE
Stormshooter
[DOD]INSANE
Zulkir
[DOD]SCOIATAEL
Onemeangreenbean
[DOD]FACEPUNCH
Xsider
[DOD]INSANE
Townsy
[DOD]HORSE RIDERS
[RS]KAL VAS FLAM
Istvan
[DOD]SCOIATAEL
Gray
Nrg
[RS]HAMMER AND SICKLE
Gurgo
[DOD]FORSAKEN
Ksardas
[DOD]INSANE
Mirk
[DOD]INSANE
Tajger
[DOD]SCOIATAEL
Scooter
Navarin
Novice Adventurer 0 / 1
You must craft 1 of 2 Beginner's Pickaxe
DESTINY BOARD
FPS: 29
Ping: 131
16:41 (UTC)
N
W
E
S
BEARSFAST

Agenda

- Deterministic Simulation – A short reminder
- How RTS-style games use it
- How MMO-style games can still use it!
- The pitfalls: How to do it and what to avoid
- A few tricks with deterministic randomness
- A few examples from Albion Online

Gameboy Multiplayer

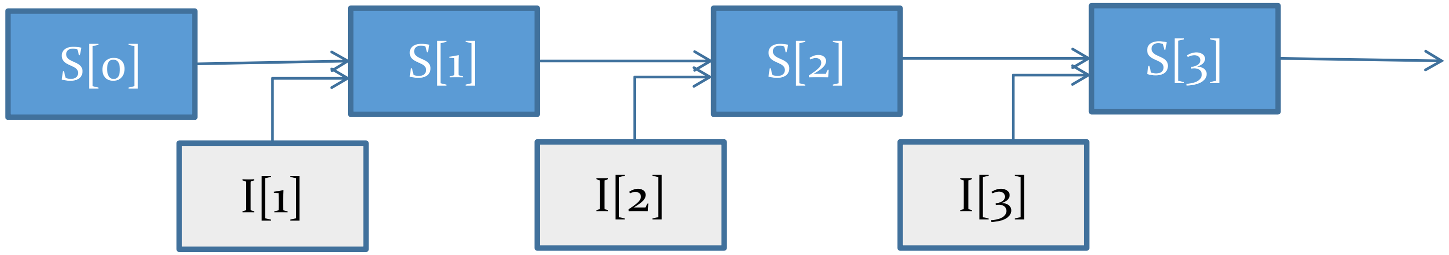
- Link cable had very limited throughput
- … as in: a few bytes per frame and player
- Syncing complex game state is impossible

- Instead: used like a controller cable! Deterministic simulation on all devices
- Frame updates are synced (effectively „lock-stepping“)
- Still used on DSi and 3DS

Deterministic Simulation?

- This should be an old hat, but…
- Deterministic: same input → same output

- Input[i] × State[i-1] = State[i]
  - where i is the simulation step number



- Given State[0] and same sequence of inputs Input[1..n]
- … all clients will produce same Sequence State[1..n]

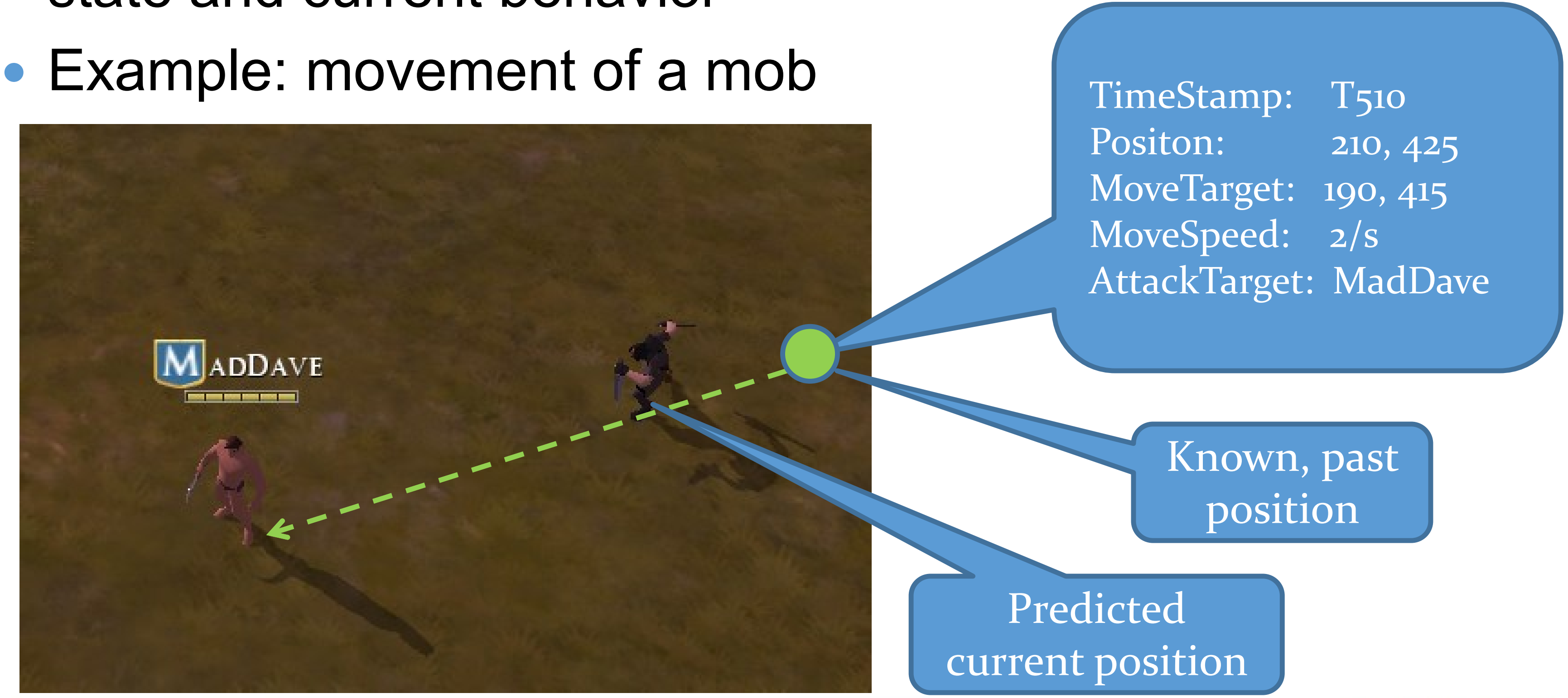
Deterministic Simulation!

- This is cool because:
  - Only need to send State[0] and Inputs through network!
    - Only Inputs if State[0] is known
  - Can save replays by saving only Inputs!
  - You can debug replays of bugs!

- Difficulties:
  - one mistake and the clients „desync“
  - must be independent of frame/thread timings
  - requires lock-stepping for online games
  - Late join requires you to send State[n]

Deterministic Simulation vs. Dead Reckoning

- Dead Reckoning:
  - Extrapolate future state of an object based on a known state and current behavior
  - Example: movement of a mob

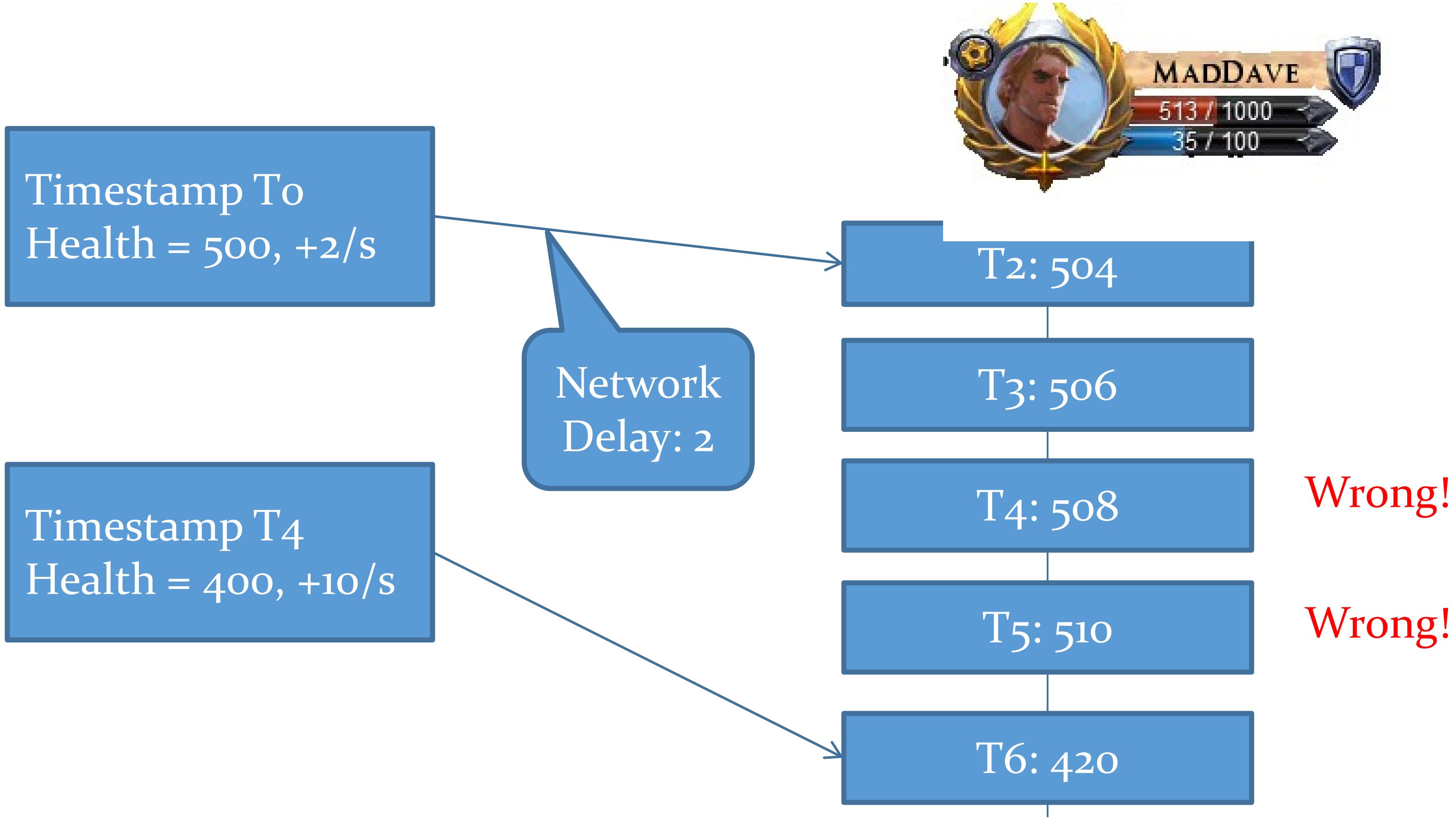


TimeStamp:	T510
Positon:	210, 425
MoveTarget:	190, 415
MoveSpeed:	2/s
AttackTarget:	MadDave

Known, past position

Predicted current position

Deterministic Simulation vs. Dead Reckoning



Timestamp To
Health = 500, +2/s

Network Delay: 2

Timestamp T4
Health = 400, +10/s

T2: 504

T3: 506

T4: 508 Wrong!

T5: 510 Wrong!

T6: 420

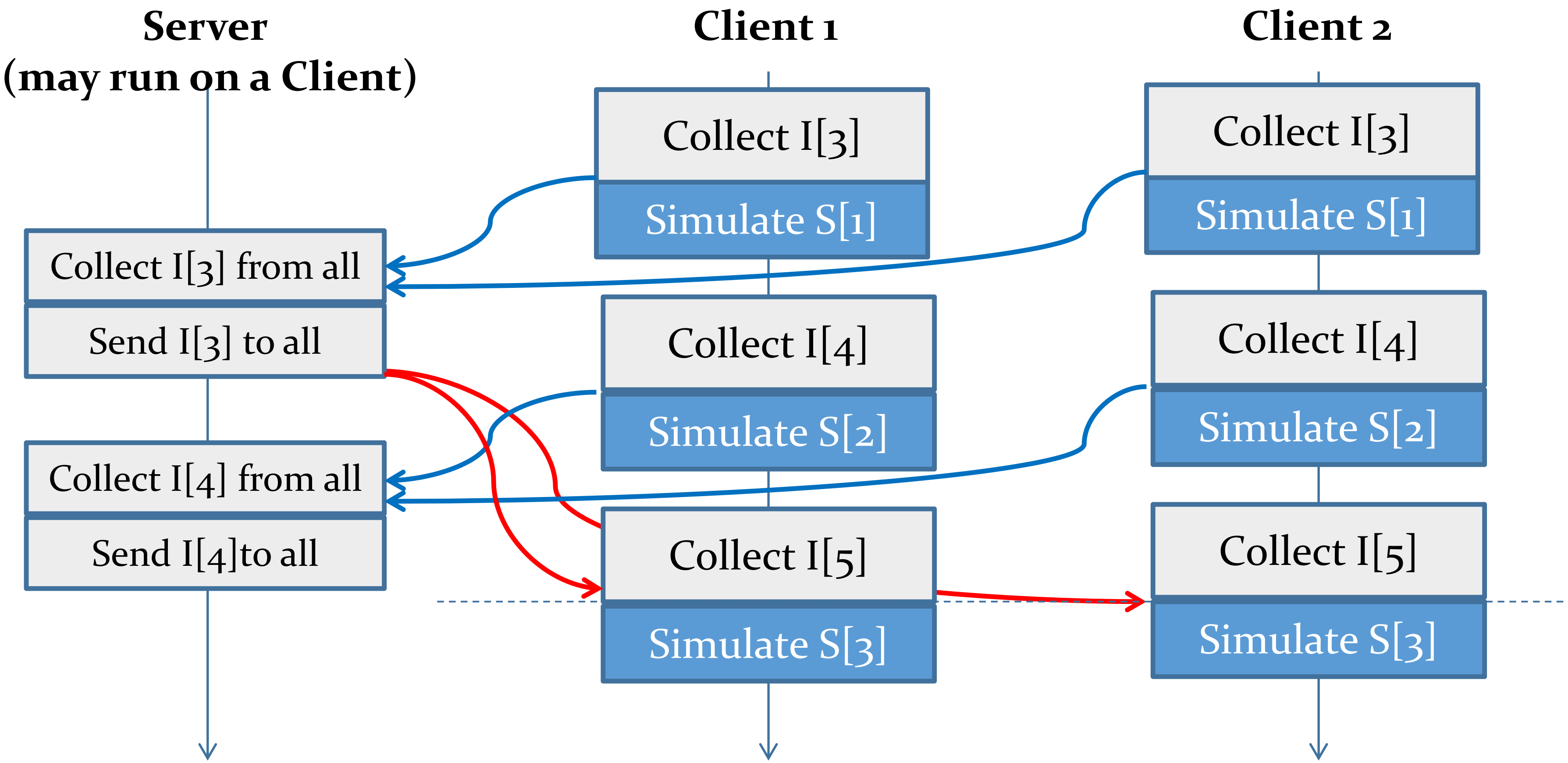
- But: this is only a prediction! May be incorrect and client may act on incorrect info!
- May have to correct state given new information!

Lock-stepping (1)

- This is how RTS games do it
- Basically everything from Age of Empires to Starcraft2

- Collect input from all players, send it to all players
  - Simulation step i can only happen when input from all players for step i has arrived (stepping is „synced“ or „locked“
  - Collect input a little earlier to account for ping
- Allows high unit count with super-small bandwidth!

Lock-stepping (2)



Lock-stepping (3)

- Problems:
  - Slowest player’s ping will be felt by all players
    - Worst case: „waiting for player“
  - Input delay is noticeable
    - Usually covered by animation, audio prompt etc.
  - Difficult to handle drop-out / late join
  - → only suitable for very limited number of players!

Actor-based determinism (1)

- Lock-stepping is not suitable for MMOs!
  - Cannot wait for players (worst ping = everyone's ping!)
  - Single player cannot „see“ full game state (just too big)
  - Everyone does a „late Join“

- BUT: can still use deterministic simulation for a single actor
  - … as long as behavior depends only on actor itself
  - example: roaming behavior of a mob (later)
  - Can mix with dead reckoning
  - Also great for visual stuff w/o gameplay influence

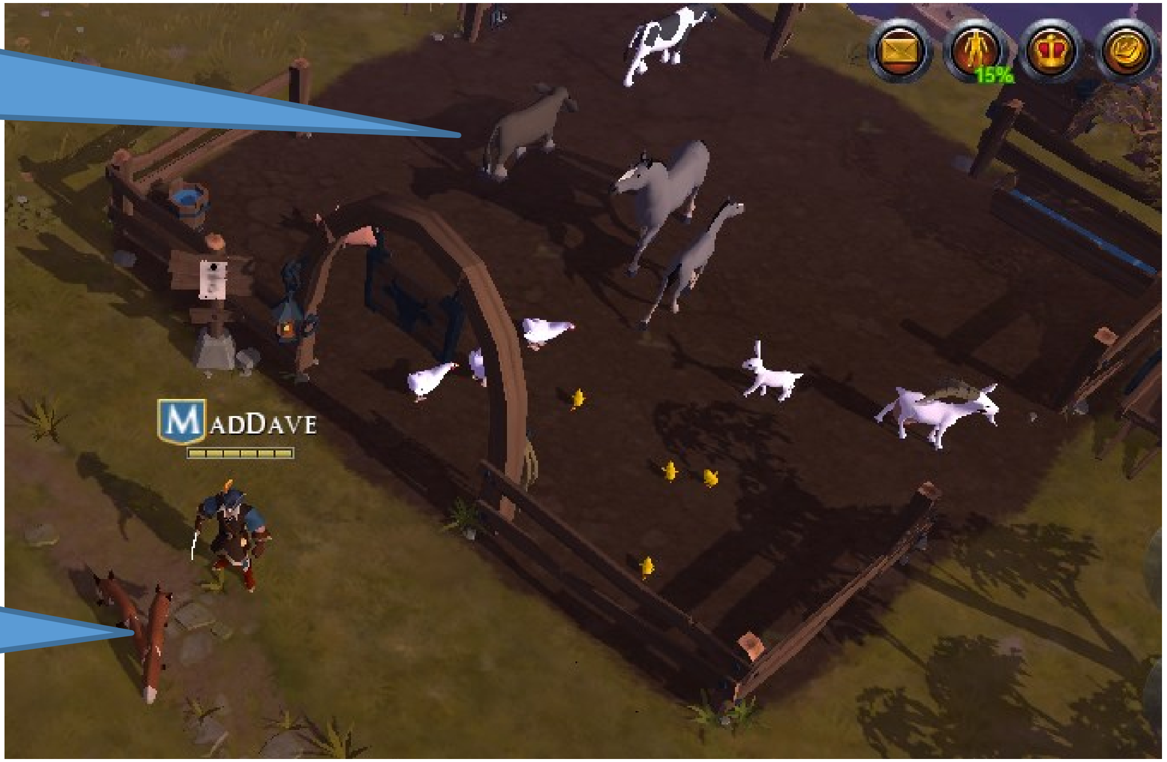
Actor-based determinism (2)

Farm Animal

- roaming around
- just eye candy!

MOBs

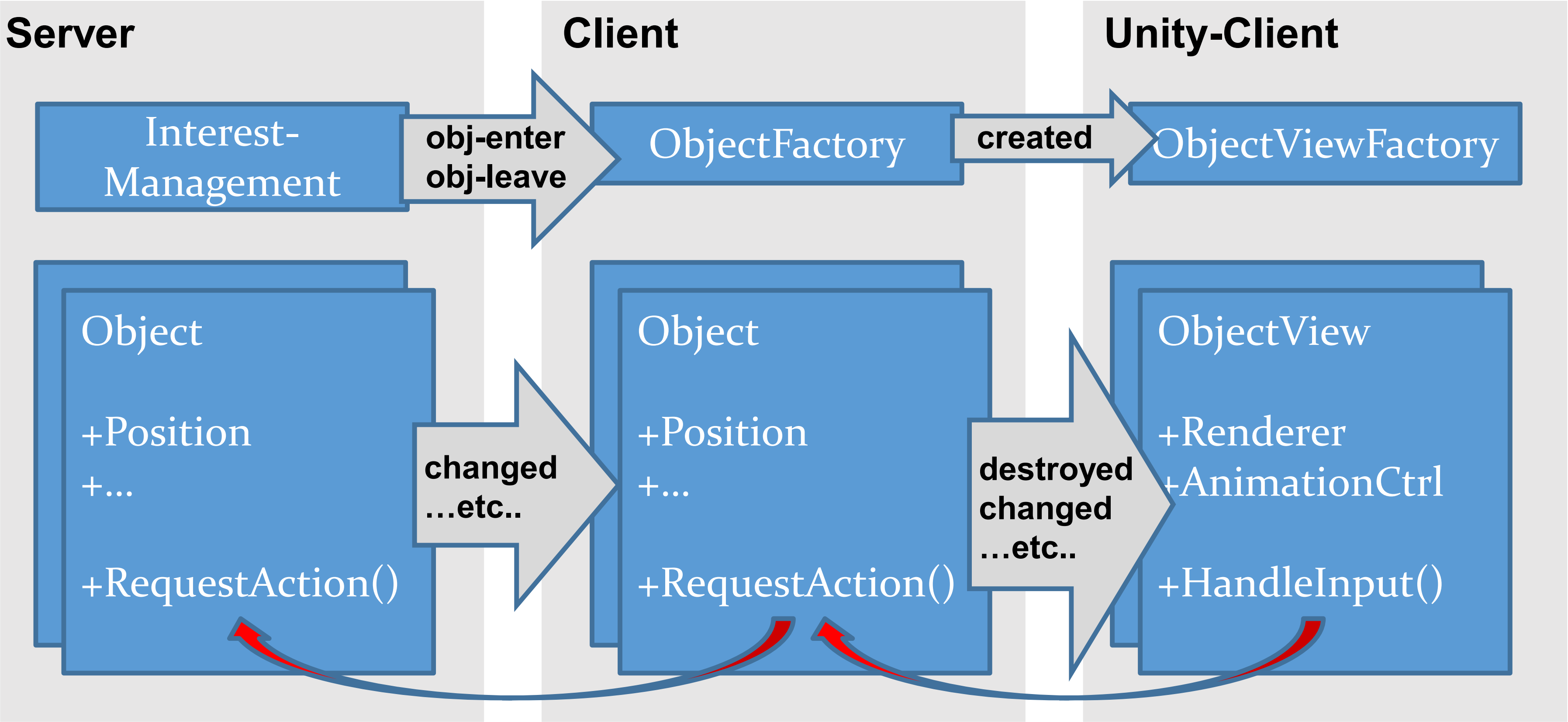
- gameplay relevant
- attackable
- can attack
- roaming around



Pitfalls & Common Mistakes

- Uninitialized variables, dangling pointers etc.
  - add an unwanted random element to the simulation
- Undefined behavior of C++ or library functions
  - Random number generators behave differently across library versions! (Roll your own!)
- Use fixed simulation timing!
  - simulation MUST NOT depend on frame timing
  - but rendering, animation MUST…
  - Need a clean **separation of simulation and presentation**

Separation


Server
Client
Unity-Client
Interest-Management
obj-enter
obj-leave
ObjectFactory
created
ObjectViewFactory
Object
+Position
+…
+RequestAction()
changed
…etc..
Object
+Position
+…
+RequestAction()
destroyed
changed
…etc..
ObjectView
+Renderer
+AnimationCtrl
+HandleInput()

The trouble with float (1)

- IEEE standard: only +, –, *, /, sqrt guaranteed to give same results everywhere
- not: sin, cos, tan etc. (different on different CPU types)

- CPU can store numbers in float or double format
- how intermediate results are stored is often unspecified (depends on compiler)

- x86: per-thread settings for precision, exceptions, rounding, denormal support
- … check the manual of your target CPUs…
- different feature sets (SIMD sets like MMX, SSE etc.)

The trouble with float (2)

- You can make floats **work** if…
  - … you stick to +, –, *, /, sqrt (write the rest yourself)
  - … you can configure compiler behavior (intermediate precision, instruction set used)
  - … you can control CPU behavior (precision, rounding etc.)
  - Best: one target CPU type, same binary for all clients

- You are in **trouble** if…
  - … you need to support a JIT environment
  - … you need to target different CPUs
  - … you need to use different compilers

Fixed Point numbers (1)

- Idea: create fractional number type based on integers
- … and use only this in (deterministic) simulation
- again: clean separation of gameplay / rendering is important

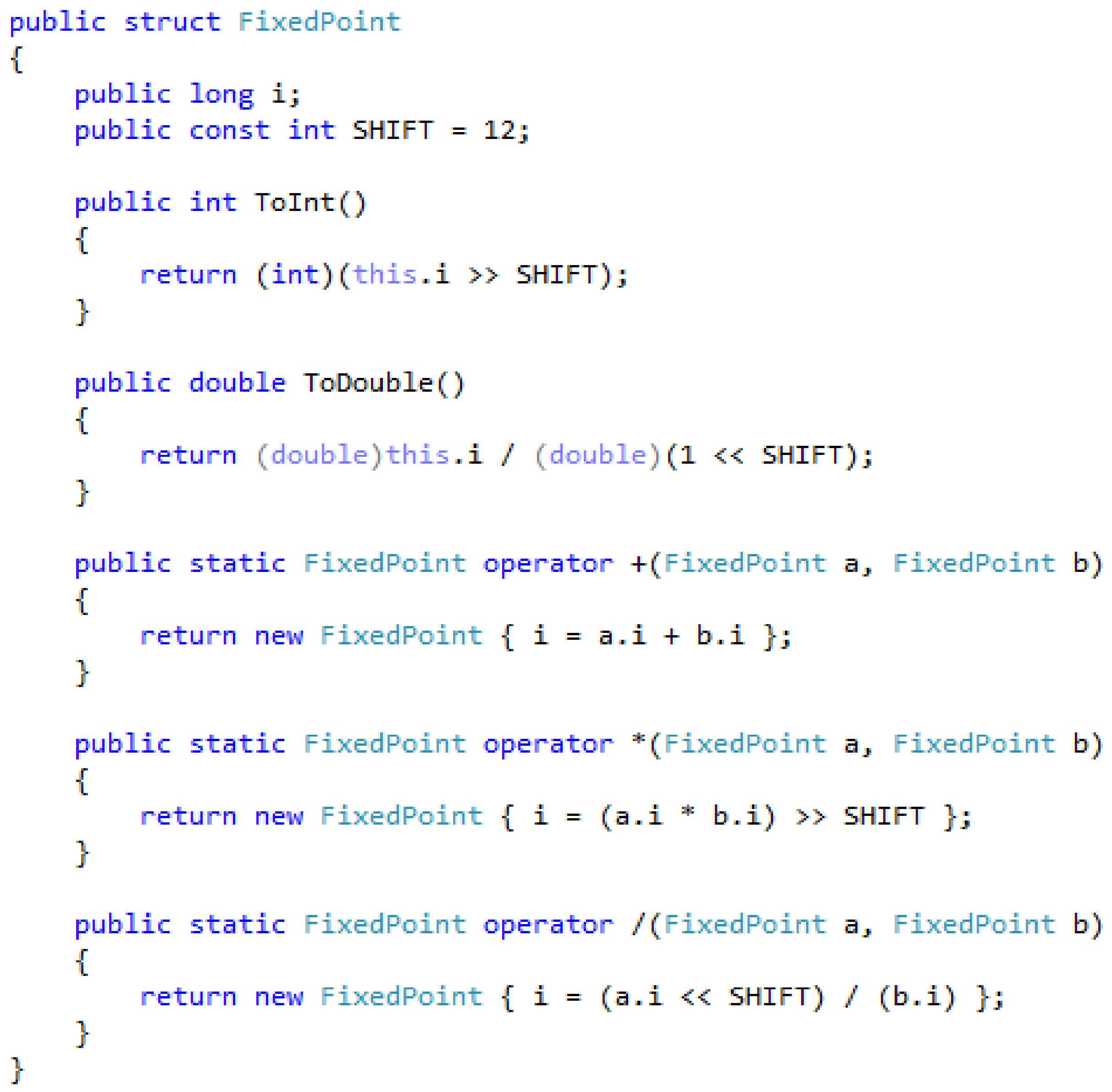
$$Z = b_m b_{m-1} \dots b_0, b_{-1} b_{-2} \dots b_{-n} = \sum_{i=-n}^{m} b_i \cdot 2^i \qquad m, n \in \mathbb{N} \quad b_i \in \{0, 1\}$$

- e.g. 110.010

  $= 1*2^2 + 1*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2} +0*2^{-3}$

  $= 1*4 + 1*2 + 0*1 + 0*0{,}5 + 1*0{,}25 + 0*0{,}125$

  $= 6{,}25$



```
public struct FixedPoint
{
    public long i;
    public const int SHIFT = 12;

    public int ToInt()
    {
        return (int)(this.i >> SHIFT);
    }

    public double ToDouble()
    {
        return (double)this.i / (double)(1 << SHIFT);
    }

    public static FixedPoint operator +(FixedPoint a, FixedPoint b)
    {
        return new FixedPoint { i = a.i + b.i };
    }

    public static FixedPoint operator *(FixedPoint a, FixedPoint b)
    {
        return new FixedPoint { i = (a.i * b.i) >> SHIFT };
    }

    public static FixedPoint operator /(FixedPoint a, FixedPoint b)
    {
        return new FixedPoint { i = (a.i << SHIFT) / (b.i) };
    }
}
```

Deterministic Randomness

- Random number generators are deterministic
  - Provided same initial seed, will produce same random sequence
- Many copy-paste-ready implementations exist
  - E.g. Mersenne Twister, WELL, XORshift
  - (Wikipedia has a list!)

- Watch out for:
  - period length
  - memory footprint
  - speed
  - warmup period
- But can we „seek“ inside the random sequence?

Cryptographic Hashes

- Cryptographic Hash functions can be used as random number sources!

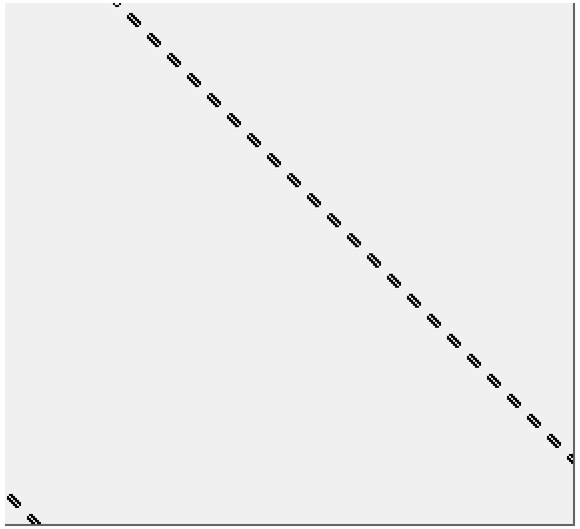
- Hash Function: converts data into unique integer
  - i.e. byte[] → int
  - … seeks to avoid „collisions“ (i.e. different data should produce hash; meaning equal distribution of hashes)
- Cryptographic hash function: not easily reversible
  - i.e. output must appear random!

Seekable random sequences

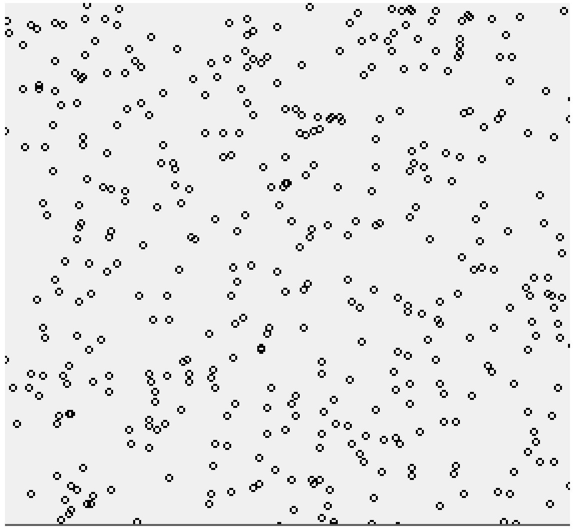
- Cryptographic Hashes can be used to build seekable random number generators!
  - because Hash(i) is random, even if i = {0,1,2,3,4,5… n}

- Note: you can do this with timestamps, coordinates, … anything really!

Seekable random sequences



Adler32

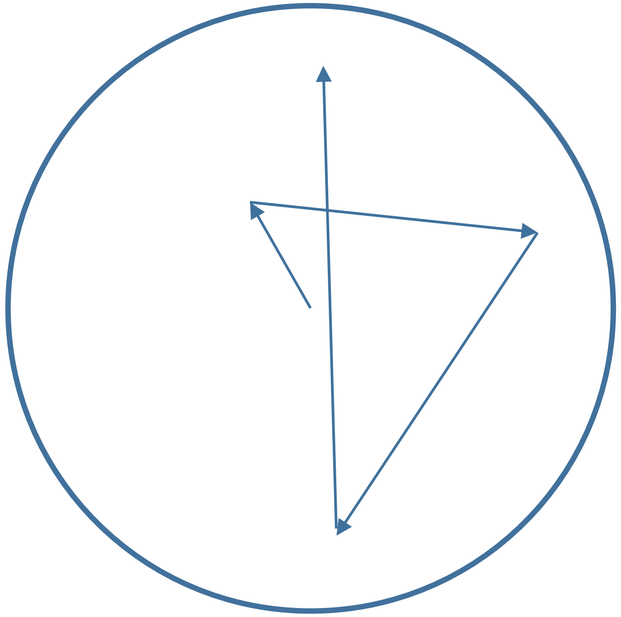


MD5

```
for(int i=0; i<1000; i+=2)
{
    DrawPoint(Hash(i) % 300, Hash(i+1) % 300)
}
```

Example: Mob Roaming Behavior

- given:
  - Mob „Home“ position
  - Roaming Radius
- repeat:
  - pick random point inside roaming circle
  - walk to random point (stop if path is blocked)
  - wait for random time (between a given min and max)

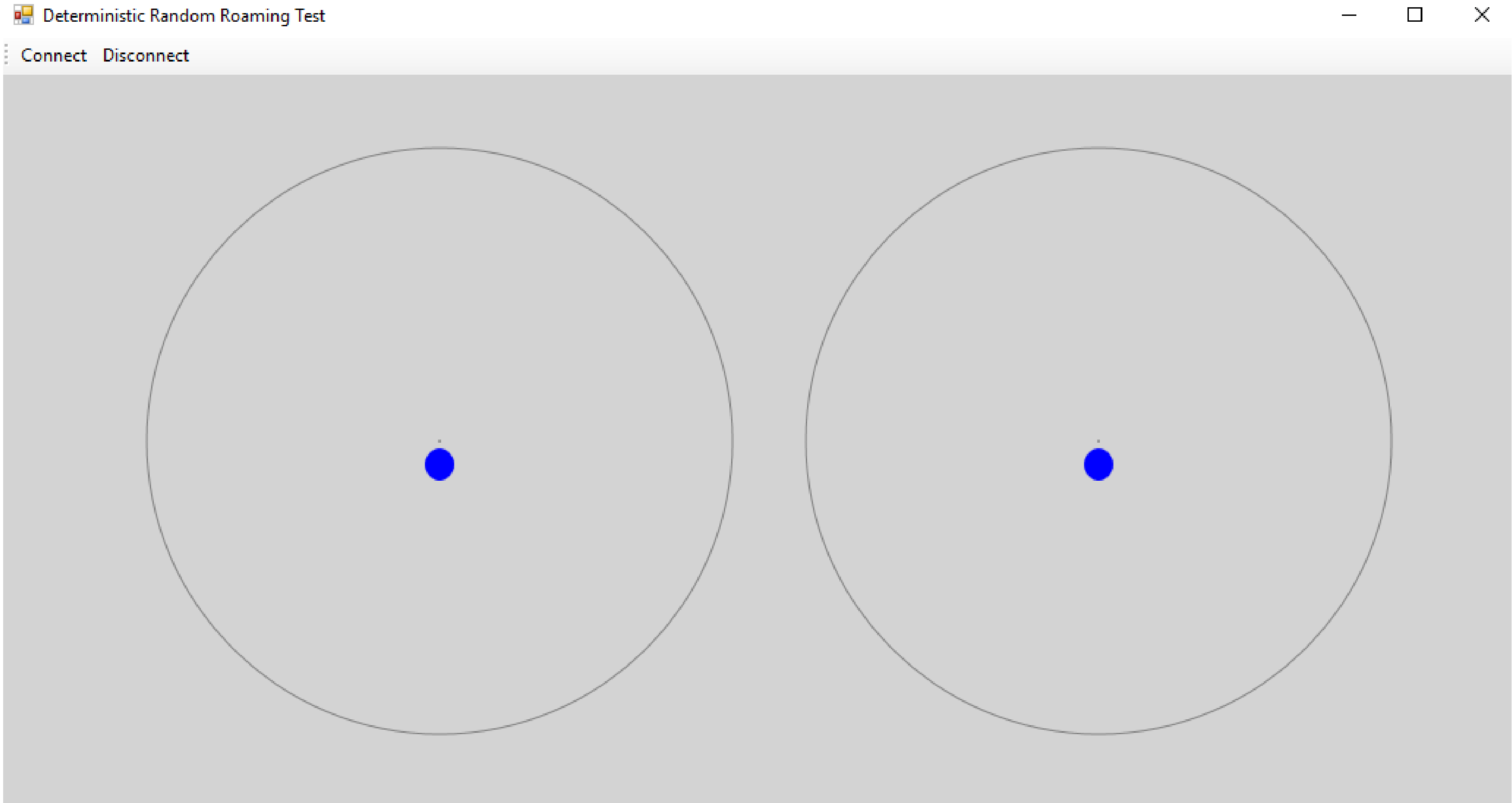


```
StartNextCycle(startTimeStamp, startPosition)
{
    init RNG with startTimeStamp
    pick „random“ moveTarget point
    if(there is a collision on the way there)…
            … the collision point is the moveTarget
    calculate the walkTime to moveTarget
    pick a random waitTime
    endTimeStamp = startTimeStamp + walkTime + waitTime
}

Render(nowTimeStamp)
{
    while(nowTimeStamp > endTimeStamp)
            StartNextCycle(endTimeStamp, moveTarget)

    if(nowTimeStamp < startTimeStamp + walkTime)
        position = LERP(startPosition, moveTarget)
    else
        position = moveTarget
}
```

Live Demo!



Takeaway

- Deterministic Simulation can greatly reduce network traffic in online/multiplayer games
- RTS-style games use fully deterministic gameplay with lock-stepping
- MMO-style games can still use actor-based deterministic simulation
- May have to use fixed point instead of float
- Hash functions are great for „randomness“ (including seekable random sequences!)

References

- **1500 Archers on a 28.8**
  - http://www.gamasutra.com/view/feature/131503/1500_archers_on_a_288_network_.php
- **Floating-Point Determinism**
  - https://randomascii.wordpress.com/2013/07/16/floating-point-determinism/
- **List of random number generators**
  - https://en.wikipedia.org/wiki/List_of_random_number_generators

Thank you!

Questions / Comments?

david@sandbox-interactive.com

We are hiring!

https://albiononline.com/en/jobs

SANDBOX INTERACTIVE

Albion ONLINE